# Results

## Lax Wendroff scheme

## Mac Cormack Scheme

## Second Order Upwind

## Runger Kutta Method

## Maximum CFL number for each scheme

Below is the maximal CFL number for each scheme and a littler larger value is tried to inspect if the scheme is table or not. Only do the test for n=1.

**Table** the maximal and tested CFL number

|  |  |  |
| --- | --- | --- |
|  | maximal value | Value tried |
| Lax Wendroff | 1 | 1.2 |
| Mac Cormack | 1 | 1.2 |
| Second Order Upwind | 2 | 2.2 |
| Runger Kutta | 2.8 | 3 |

## Conclusion

* For n=1 all the schemes can have almost same results while for n=3 variance in the phrase and magnitude is shown from figure.
* For n=1 little magnitude decreasing is shown with the decreasing of the CFL while for n=3 larger variance is shown.
* When the CFL is a little larger than the maximal CFL the results from Lax Wendroff and Mac Cormack scheme shows fluctuation at some points which indicates that the instability of the scheme.
* When CFL is a little larger than the maximal CFL the results from second order upwind scheme and Runger Kutta scheme coincides with the benchmark. This probably indicates that compared to the former two schemes the latter two scheme is relatively robust.

# Codes

## Main

Program solve\_wave\_equation

implicit none

C.. define variable

integer i,j, k, n,imax, count\_no

integer case\_no

real dx, lx(100), dt, t

real pi

real cfl

real c

real u(100,100)

real u\_ana(100)

character(len=100) filename

C.. parameter initialization for control

!parameter (case\_no=2)

!parameter (cfl=0.3)

!parameter (n=1)

parameter (imax=41)

C.. parameter initialization

parameter(dx=1.0)

parameter (c=1)

parameter (pi=3.1415)

C.. variable initialization

i=1

j=1

lx(1)=0.0

t=0.0

count\_no=1

C.. read the parameters

write (6,\*) "Case\_no=?"

read (5,\*) case\_no

write (6,\*) "n=?"

read (5,\*) n

write (6,\*) "cfl=?"

read (5,\*) cfl

C.. This is the 8th assignment of CFD course

write (6,\*) 'This is the 8th assignment of CFD course'

C.. space discretization

do i=2,41

lx(i)=lx(i-1)+dx

end do

C.. give initial value

do i=1,41

u(i,1)=sin(2\*n\*pi\*lx(i)/40)

end do

C.. use CFL and dx to calculate dt

dt=cfl\*dx/c

C.. Routine for different schemes

select case (case\_no)

C.. 1--Lax Wendroff one step scheme

case (1)

write (6,\*) 'Entrance for the Lax-Wendroff one step method'

call lax\_wendroff (u,lx,imax,c,dx,dt,t,j)

C.. 2--MacCormack Two Step scheme

case (2)

write (6,\*) 'Entrance for the MacCormack Two Step scheme'

call Mac\_Cormack (u,lx,imax,c,dx,dt,t,j)

C.. 3--Second order upwind scheme

case (3)

write (6,\*) 'Entrance for the Second order upwind scheme'

call second\_order\_upwind (u,lx,imax,c,dx,dt,t,j)

C.. 4--4th order Runge-Kutta scheme

case (4)

write (6,\*) 'Entrance for 4th order Runge-Kutta scheme'

call runger\_kutta (u,lx,imax,c,dx,dt,t,j)

C.. default to output error information

case default

write (6,'(A)') 'can not recognize the selector i'

end select

C.. delete me

C.. Output data

write (filename,\*) "count",".txt"

open (1, file='results.txt', status='unknown')

!open (1, file='wave\_equ.txt', status='unknown', access='append')

if (case\_no==1) then

write(1,\*) 'results from Lax-Wendroff one step scheme'

else if (case\_no==2) then

write(1,\*) 'results from MacCormack two step scheme'

else if (case\_no==3) then

write(1,\*) 'results from second order upwind scheme'

else if (case\_no==4) then

write(1,\*) 'results from 4th order Runge-Kutta scheme'

else

write (1,\*) 'Error with outputting data'

end if

write (1,\*) 'cfl=', cfl, 'n=',n, 'time step size=', dt

write (1,2001)

2001 format (5x,'X',10x,'Numerical solution',5x,'Analytical solution')

do i=1,41

u\_ana(i)=sin(2\*n\*pi\*(lx(i)-18)/40)

write (1,2002) lx(i),u(i,j ), u\_ana(i)

2002 format (f13.10,6x, f13.10, 10x, f13.10)

end do

count\_no=count\_no+1

End program solve\_wave\_equation

## Subroutine Lax\_wendroff

subroutine lax\_wendroff (u,lx,imax,c,dx,dt,t,j)

C.. this is the subroutine for lax wendroff method

implicit none

real u(100,100)

real lx(100)

integer imax

real c,dx,dt

real u\_temp(100,100)

integer i,j,k

real t

C.. variable initialization

i=1

j=1

k=1

t=dt

C.. main entrance for lax wendroff method

do while (t<=18)

do i=1,41

! Special treatment of point 1

if (i==1) then

u(i,j+1) = u(i,j)

> - c\*dt/2/dx\*(u(i+1,j)-u(imax-1,j))

> + c\*c\*dt\*dt/2/dx/dx\*(u(i+1,j)-2\*u(i,j)

> + u(imax-1,j))

! Special treatment of point 41

else if (i==41) then

u(i,j+1) = u(i,j)-c\*dt/2/dx\*(u(2,j)-u(i-1,j))

> + c\*c\*dt\*dt/2/dx/dx\*(u(i-1,j)-2\*u(i,j)

> + u(2,j))

! the regular point

else

u(i,j+1) = u(i,j)-c\*dt/2/dx\*(u(i+1,j)-u(i-1,j))

> + c\*c\*dt\*dt/2/dx/dx\*(u(i+1,j)-2\*u(i,j)

> + u(i-1,j))

end if

end do

j=j+1

t=t+dt

end do

return

end

## Subroutine Mac\_Cormack

subroutine Mac\_Cormack (u,lx,imax,c,dx,dt,t,j)

C.. this is the subroutine for the Mac Cormack method

implicit none

real u(100,100)

real lx(100)

integer imax

real c,dx,dt

real u\_temp(100,100)

integer i,j,k

real t

C.. variable initialization

i=1

j=1

k=1

t=dt

C.. main entrance for Mac Cormack method

do while (t<=18)

! first step-prediction

do i=1,imax

if (i==imax) then

u\_temp(i,j+1)=u(i,j)-c\*dt/dx\*(u(2,j)-u(i,j))

else

u\_temp(i,j+1)=u(i,j)-c\*dt/dx\*(u(i+1,j)-u(i,j))

end if

end do

! second step prediction

do i=1,imax

if (i==1) then

u(i,j+1)=0.5\*(u(i,j)+u\_temp(i,j+1)

> -c\*dt/dx\*(u\_temp(i,j+1)-u\_temp(imax-1,j+1)))

else

u(i,j+1)=0.5\*(u(i,j)+u\_temp(i,j+1)

> -c\*dt/dx\*(u\_temp(i,j+1)-u\_temp(i-1,j+1)))

end if

end do

t=t+dt

j=j+1

end do

return

end

## Subroutine Second\_order\_upwind

subroutine second\_order\_upwind (u,lx,imax,c,dx,dt,t,j)

C.. this is the subroutine for the second order upwind method

implicit none

real u(100,100)

real lx(100)

integer imax

real c,dx,dt

real u\_temp(100,100)

integer i,j,k

real t

C.. variable initialization

i=1

j=1

k=1

t=dt

C.. main entrance for second order upwind method

do while (t<=18)

!first step-predictor

do i=1,imax

if (i==1) then

u\_temp(i,j+1)=u(i,j)-c\*dt/dx\*(u(i,j)-u(imax-1,j))

else

u\_temp(i,j+1)=u(i,j)-c\*dt/dx\*(u(i,j)-u(i-1,j))

end if

end do

!second step-corrector

do i=1,imax

if (i==1) then

u(i,j+1)=0.5\*(u(i,j)+u\_temp(i,j+1)-c\*dt/dx\*(u\_temp(i,j+1)

> -u\_temp(imax-1,j+1))-c\*dt/dx\*(u(i,j)

> -2\*u(imax-1,j)+u(imax-2,j)))

else if (i==2) then

u(i,j+1)=0.5\*(u(i,j)+u\_temp(i,j+1)-c\*dt/dx\*(u\_temp(i,j+1)

> -u\_temp(i-1,j+1))-c\*dt/dx\*(u(i,j)

> -2\*u(i-1,j)+u(imax-1,j)))

else

u(i,j+1)=0.5\*(u(i,j)+u\_temp(i,j+1)-c\*dt/dx\*(u\_temp(i,j+1)

> -u\_temp(i-1,j+1))-c\*dt/dx\*(u(i,j)

> -2\*u(i-1,j)+u(i-2,j)))

end if

end do

t=t+dt

j=j+1

end do

return

end

## Subroutine Runger\_Kutta

subroutine runger\_kutta (u,lx,imax,c,dx,dt,t,j)

C.. this is the subroutine for Runger-Kutta method

implicit none

real u(100,100)

real lx(100)

integer imax

real c,dx,dt

real u\_temp1(100), u\_temp2(100),u\_temp3(100)

real R1(100), R2(100), R3(100), R4(100)

integer i,j,k

real t

C.. variable initialization

i=1

j=1

k=1

t=dt

C.. main entrance for Runger-Kutta method

do while (t<=18)

! first step

! calculate R1

do i=1,imax

if (i==1) then

R1(i)=-1\*c\*(u(i+1,j)-u(imax-1,j))/(2\*dx)

else if (i==41) then

R1(i)=-1\*c\*(u(2,j)-u(i-1,j))/(2\*dx)

else

R1(i)=-1\*c\*(u(i+1,j)-u(i-1,j))/(2\*dx)

end if

end do

! calculate u\_temp1

do i=1,imax

u\_temp1(i)=u(i,j)+0.5\*dt\*R1(i)

end do

! Second step

! calculate R2

do i=1,imax

if (i==1) then

R2(i)=-1\*c\*(u\_temp1(i+1)-u\_temp1(imax-1))/(2\*dx)

else if (i==41) then

R2(i)=-1\*c\*(u\_temp1(2)-u\_temp1(i-1))/(2\*dx)

else

R2(i)=-1\*c\*(u\_temp1(i+1)-u\_temp1(i-1))/(2\*dx)

end if

end do

! Calculate u\_temp2

do i=1,imax

u\_temp2(i)=u(i,j)+0.5\*dt\*R2(i)

end do

! Third step

! calculate R3

do i=1,imax

if (i==1) then

R3(i)=-1\*c\*(u\_temp2(i+1)-u\_temp2(imax-1))/(2\*dx)

else if (i==41) then

R3(i)=-1\*c\*(u\_temp2(2)-u\_temp2(i-1))/(2\*dx)

else

R3(i)=-1\*c\*(u\_temp2(i+1)-u\_temp2(i-1))/(2\*dx)

end if

end do

! calculate u\_temp3

do i=1,imax

u\_temp3(i)=u(i,j)+dt\*R3(i)

end do

! calculate R4

do i=1,imax

if (i==1) then

R4(i)=-1\*c\*(u\_temp3(i+1)-u\_temp3(imax-1))/(2\*dx)

else if (i==41) then

R4(i)=-1\*c\*(u\_temp3(2)-u\_temp3(i-1))/(2\*dx)

else

R4(i)=-1\*c\*(u\_temp3(i+1)-u\_temp3(i-1))/(2\*dx)

end if

end do

! Forth Step

do i=1,imax

u(i,j+1)=u(i,j)+dt/6\*(R1(i)+2\*R2(i)+2\*R3(i)+R4(i))

end do

t=t+dt

j=j+1

end do

return

end